Identifying Test Requirements by Analyzing SLA Guarantee Terms

M. Palacios, J. García-Fanjul, J. Tuya
Department of Computer Science
University of Oviedo
Campus de Viesques, Asturias, Spain
{palaciosmarcos, jgfanjul, tuya}@uniovi.es

George Spanoudakis
School of Informatics
City University London
London, UK
G.E.Spanoudakis@ city.ac.uk

***Abstract*—Service Level Agreements (SLAs) are used to specify the negotiated conditions between the provider and the consumer of services. In this paper we present a stepwise method to identify and categorize a set of test requirements that represent the potential situations that can be exercised regarding the specification of each isolated guarantee term of an SLA. This identification is addressed by means of devising a set of coverage levels that allow grading the thoroughness of the tests. The utilization of these test requirements would focus on twofold objectives: (1) the generation of a test suite that allows exercising the situations described in the test requirements and (2) the support for the derivation of a monitoring plan that checks the compliance of these requirements at runtime. The approach is illustrated over an eHealth case study.**

Keywords - Service Level Agreements; Test Requirements; Software Testing; Service Monitoring.

I. INTRODUCTION AND MOTIVATION

In the context of service-oriented architectures, Service Level Agreements (SLAs) are technical documents that contain the negotiated conditions between service providers and consumers. These agreements act as a guarantee where the set of terms that govern the executions of the constituent services of an application are specified. They also state the penalties to be applied upon the violation of such terms. It is therefore important for both stakeholders to avoid or minimize the consequences derived from SLA violations.

Currently, most research uses monitoring techniques in order to detect SLA violations in service based applications (SBAs) at runtime [17]. In complex and critical scenarios, these approaches require spending a considerable amount of effort and cost with the aim of deriving a suitable plan to observe the potential error-prone situations and check the compliance of services with the SLA. In addition, these reactive approaches are useful in detecting problems in SBAs although such problems are detected after they have already occurred and, therefore, any further consequences that they might have cannot be avoided. Thus, proactive approaches that are aimed at forecasting SLA violations or even preventing them are also being proposed [12].

In previous work [18], we presented a general method to test SLA-aware service based applications, exploiting the benefits of both proactive and reactive approaches in order to detect problems in SBAs. Aligned with this testing method, in this paper we focus on the identification of test requirements with the aim of: (1) generating a test suite that exercises such requirements (proactive approaches) and (2) deriving a monitoring plan that allows checking whether these requirements are exercised at runtime (reactive approaches). Both approaches present different characteristics that make them complementary to increase the confidence in the correct behavior of the application.

On the one hand, some of the identified test requirements can be exercised in a pre-production or controlled testing environment. To achieve this, a set of test cases must be designed trying to cover as many test requirements as possible. With the execution of these test cases, we can anticipate problems in the SBA and take proactive measures to avoid or mitigate SLA violations and their corresponding consequences. On the other hand, test requirements may also be used to guide the monitoring plan in order to decide the specific situations that have to be observed at runtime, when the services are already deployed in the operational environment.

In this paper we address the identification of test requirements in service based applications using the information represented in SLAs. The contributions of this paper are summarized as follows.

(a) We devise a set of coverage levels regarding the SLA that allow grading the thoroughness of the tests.
(b) Focusing on the first of these levels, we define a test criterion in order to identify and categorize a set of test requirements, which represent different situations that are interesting to test or monitor for the given SLA. The categorization allows establishing a prioritization according to the testing objective.
(c) The identification and categorization of these test requirements are illustrated over a case study.

The content of the paper is organized as follows. Section II presents a general overview of the approach. Section III describes a logic that considers the potential evaluation values of a Guarantee Term. Section IV describes how test requirements can be identified and categorized in the first coverage level. Section V illustrates the application of the approach in a case study. Section VI outlines the state of the research in the addressed topic. Finally, Section VII summarizes the conclusions and the future work.

II. General Overview

In the context of software testing, *test requirements* are specific features and situations of the Software Under Test (SUT) that must be satisfied or covered during testing [14]. The task of identifying test requirements is usually performed by means of the application of a *test criterion*. The definition of *coverage levels* allow grading how exhaustive the identification of test requirements can be. In some scenarios it could be possible to design an in-depth test suite which involves a high cost in terms of money or effort. However, in other situations there may be constraints that hinder the definition and execution of tests and force the tester to select a less exhaustive coverage level or prioritize the tests according to the coverage level. The selection of a coverage level always tries to maximize the trade-off among different criteria such as cost, benefit or risks.

In SBAs, an SLA specifies a set of terms that are logically combined into a hierarchical structure by means of compositor elements. Hence, test requirements can be identified based on different information represented in the agreement. Thus, we have defined three different coverage levels regarding the SLA:

- Guarantee Term Coverage Level
- Compositor Coverage Level
- SLA Coverage Level

Guarantee terms can be considered as the most indivisible condition of an SLA so a first coverage level named *Guarantee Term Coverage Level* is used to represent all the potential test requirements regarding the specification of each isolated guarantee term. In addition to this information, more exhaustive test requirements can be achieved applying a test criterion to the logical conditions represented in the compositor elements. These new test requirements belong to the *Compositor Coverage Level*. Finally and considering the specification of the SLA as a whole with all its atomic and logical conditions, a global *SLA Coverage Level* can be defined, which represents all the situations that are interesting to test according to the content of such SLA.

In addition to this and disregarding whether the test requirements have been identified according to one coverage level or other, such test requirements are later exercised through the derivation of a suitable test suite. Typically, the extent to which a test criterion is satisfied by a test suite is measured in terms of *coverage* which can be defined as the percent of test requirements that are exercised. The generation of test cases is performed with the aim of obtaining the most cost effective set that fulfils the expected coverage. Thus, typically a test case can cover many test requirements. Currently, we are focusing on the identification of test requirements so the derivation of test cases is out of scope of this paper.

In order to define different test criteria with the aim of identifying test requirements, we have devised a logic that allows evaluating each of the internal elements of an SLA Guarantee Term. Although these terms can be described using any of the multiple languages that have been previously proposed, our approach uses WS-Agreement [1] because it is a well-accepted standard for the management of SLAs. We tackle the description of this logic in the following section and the criteria to identify the test requirements in the remainder of the paper.

III. Evaluation of SLA Guarantee Terms

One of the most important tasks in the management of SLAs is the evaluation of the terms included in the agreement. This evaluation requires checking the specification of the terms and their internal elements and making a decision about the fulfillment of such terms.

Specifically, in WS-Agreement a Guarantee Term (GT) contains a Scope that specifies the list of the services and, optionally, a substructure of a service (for example, a particular method or end point) the term applies to, a Qualifying Condition (QC) which is an assertion that indicates whether the term is valid or not, and the Service Level Objective (SLO) which is the guarantee that must be met. Optionally, the penalty for not having satisfied the guarantee can be specified in the Business Value List (BVL) of the term.

Given the syntax of a Guarantee Term, after analyzing the collected information from the service executions at runtime a guarantee term can be evaluated as:

- *Fulfilled* if and only if the methods of the services specified in the Scope have been executed, the Qualifying Condition has been met and the Service Level Objective has been satisfied.
- *Violated* if and only if the methods of the services specified in the Scope have been executed, the Qualifying Condition has been met and the Service Level Objective has not been satisfied.

Typically, the evaluation of a guarantee terms is performed using a binary logic that indicates whether the term has been fulfilled or not. However, from a testing point of view, this two-value logic may not be enough to evaluate all the potential situations derived from the guarantee term. For example, situations where the methods of the services associated to a guarantee term have not been executed should be analyzed as well within the evaluation process. Considering such cases introduces the need for an additional evaluation value, under which:

- A guarantee term is evaluated as *Not Determined* if and only if the methods of the services specified in the Scope have not been executed and the Qualifying Condition is met.

Actually, WS-Agreement identifies these three situations as the potential runtime states of an SLA. However, analyzing the internal elements of a Guarantee Term and its interpretation according to the standard, we have to consider a new situation where the term is not evaluated with any of the three aforementioned values and which has not been explicitly identified in WS-Agreement. This situation arises when the *Qualifying Condition* of the term is not met during the execution of services. In this case, the Guarantee Term becomes invalid and it must not

be taken into account for the purpose of the evaluation of the SLA so:

- A guarantee term is evaluated as *Inapplicable* if and only if the Qualifying Condition has not been satisfied.

Hence, a Guarantee Term denoted by *t* can be evaluated using a function *ev*, which can provide four different values as output:

ev(t) = { Fulfilled (F), Violated (V), Not Determined (ND), Inapplicable (I)}

Guarantee Terms in WS-Agreement can be logically combined into a hierarchical structure using the specific compositor elements All, OneOrMore and ExactlyOne (equivalent to AND, OR and XOR logical operators respectively). However, the evaluation of these compositors does not affect the identification of test requirements within the *Guarantee Term Coverage Level* so the remainder of this paper focuses on the evaluation of individual Guarantee Terms.

At this point, we are assuming that the SBA is our SUT and the evaluation process is performed once in a specific point in time after the execution of the services and the guarantee condition affects just one execution of the SUT but not multiple executions.

IV. SLA Test Requirements

In this section, we are focusing on the identification and categorization of a set of test requirements that represent situations in the *Guarantee Term Coverage Level* outlined in Section II. In order to obtain such test requirements, we analyze the content of the Guarantee Terms taking Section III into account so all the potential evaluation values are exercised.

A. Identification of Test Requirements

According to the previous section, there are four different values of evaluation for a guarantee term. This implies that we would need to identify four different test requirements with the aim of satisfying all the evaluation values of a guarantee term with this logic. However, the internal syntactic structure and the semantics of a guarantee term in WS-Agreement requires a more exhaustive coverage criterion to represent all the potential situations that are interesting to observe or exercise from a testing point of view. Consider Figure 1 where the internal elements of a GT (Scope, Qualifying Condition and Service Level Objective) are represented. At the top of the figure, we check whether the methods of the services specified in the Scope have been invoked or not so this condition is evaluated with two potential values (satisfied/unsatisfied). Moreover, the content of the Qualifying Condition and the Service Level Objective represent conditions that are also evaluated as satisfied or unsatisfied. Hence, we apply all the combinations of these three internal elements of each GT. As there are three internal elements with their corresponding two truth values, we obtain eight different situations but there are two combinations that do not make sense due to the semantic meaning of the internal elements of the guarantee term. This pair of situations arises when the methods of the services specified in the Scope have not been executed so it is impossible to check whether the Service Level Objective has been fulfilled or not (right branch of the figure). Thus, we obtain a total number of six test requirements for a Guarantee Term (identified by TR1-TR6).

In detail, four of the total requirements are identified when the methods of the services specified in the Scope are invoked (left branch of Figure 1):

TR1 The methods of the services are invoked, the Qualifying Condition is satisfied and the Service Level Objective is satisfied (GT evaluated as Fulfilled).

TR2 The methods of the services are invoked, the Qualifying Condition is satisfied and the Service Level Objective is unsatisfied (GT evaluated as Violated).

TR3 The methods of the services are invoked, the Qualifying Condition is unsatisfied and the Service Level Objective is satisfied (GT evaluated as Inapplicable).

TR4 The methods of the services are invoked, the Qualifying Condition is unsatisfied and the Service Level Objective is unsatisfied (GT evaluated as Inapplicable).

In addition to these test requirements, we also consider those situations where the methods of the services specified in the Scope element have not been invoked at the time of the evaluation (right branch of Figure 1). Namely, we include what happens when the Qualifying Condition is satisfied / unsatisfied while the methods of the services are not executed. For each Guarantee Term, other two test requirements are identified as well:

TR5 The methods of the services are not executed while the Qualifying Condition is satisfied (GT evaluated as Not Determined).

TR6 The methods of the services are not executed while the Qualifying Condition is unsatisfied (GT evaluated as Inapplicable).

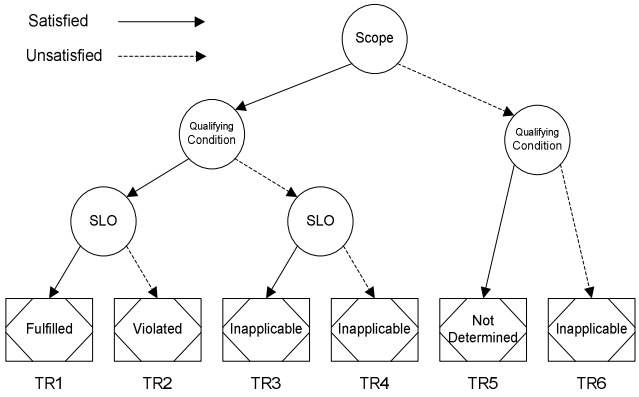

Figure 1. Combination of internal elements of a Guarantee Term

The obtaining of these six test requirements for each Guarantee Term represents the general case of the identification process. However, there are also some usual particular cases (PC) that we have to deal with.

PC1: The first particular case arises when the Guarantee Term has no Qualifying Condition associated. We have previously mentioned that the Qualifying Condition determines whether a term is valid or not so it must be considered during the evaluation process. In this particular case, the term is always valid so only three test requirements (TR1, TR2 and TR5) are identified. Furthermore, the specification of the test requirements TR1 and TR2 must be adapted as "The methods of the services are invoked and the Service Level Objective is satisfied / unsatisfied" respectively and test requirement TR5 as "The methods of the services are not executed (GT evaluated as Not Determined)".

PC2: This particular case arises when the Qualifying Condition of the Guarantee Term is an assertion over service attributes. This case occurs because the semantics of the Qualifying Condition also affect the identification of the test requirements. WS-Agreement states that the Qualifying Condition is an assertion over service attributes and/or external factors. For example, in the former this condition may make reference to an input parameter or condition of the service while in the latter it can represent a specific state of the SUT. If this particular case, the combinations performed in test requirements TR5 and TR6 do not make sense because it is impossible to check the fulfillment of the QC if the methods of the services have not been executed. In such case, test requirements TR5 and TR6 are joined in only one as "The methods of the services are not executed (GT evaluated as Not Determined)" so we would obtain one test requirement less than in the general case.

B. *Categorization of Test Requirements*

The decision about the depth of the test is pointed out by the coverage level. However, once such coverage level has been selected, different testing objectives regarding the tests can be selected with the aim of leading to the identification of test requirements. Thus, the utilization of a coverage level can be refined through the definition of different categories so as the tester has the capability to prioritize which of these objectives are going to be satisfied within that coverage level. In addition to this, the categorization of test requirements can also be used to establish monitoring objectives, making a decision about the features of the SUT that are more interesting to be observed at runtime. Hence, these categories can be used to guide the identification of specific test requirements instead of obtaining the whole set of situations from the terms of the SLA.

TABLE I. *TEST REQUIREMENTS CATEGORIES*

Cat.	Description	Test Req.
1	Expected behavior of the SUT	TR1
2	Test the behavior after a term violation	TR2
3	Testing need indicator while monitoring	TR3, TR4
3.1	Test the monitor to avoid false positives	TR4
4	Test the effects of not executing a service	TR5, TR6

Table I displays the categorization of test requirements in the Guarantee Term Coverage Level according to their meaning or testing objective. The first column of this table shows the identifier of each category. The second column outlines the description of the testing objective of the category. Finally, last column lists the test requirements that are included in such category.

Category 1 (C1) makes reference to the situations where the execution of the SUT satisfies the conditions specified in the guarantee term of the SLA so such term is evaluated as Fulfilled. From a monitoring point of view, these situations represent the expected behavior of the SUT so they should be continuously exercised if no problem arises during the period of time the system is being observed. Test requirements TR1 identified from the Guarantee Term are included in this category.

Category 2 (C2) represents those requirements that involve a violation of any of the terms included in the SLA so test requirements TR2 are included in this category. Even when an SLA violation arises, the application must deliver an expected behavior despite of any detected problem. Thus, the application will have to manage the violation according to the business values such as penalties specified in the SLA. Furthermore, the monitoring system must be able to detect the problem and report it in a proper way as well as evaluating the term as Violated. These situations are very interesting in both testing and monitoring approaches because their detection allows analyzing the information collected from the monitor and making a decision about any corrective action in order to solve the problem and avoid future consequences.

Category 3 (C3) includes those requirements that represent executions where the services are invoked under circumstances that do not satisfy the Qualifying Condition so the terms become invalid and they must not be taken into account when evaluating the SLA. While monitoring, the systematic fulfillment of these requirements means that the application is continuously being executed under the same conditions specified in the QC so we do not have evidences about how the application would behave when the execution conditions change. Hence, they indicate the need of designing tests with the aim of checking whether the application is able to fulfill the GT in the future. Test requirements TR3 and TR4 are included in this category.

Within this category, there is a subcategory 3.1 (C3.1) of requirements that can be used to check the behavior of the monitoring system that gathers information from the executions of the services and makes a decision about the evaluation of the SLA. More specifically, these requirements aim at checking that this monitor does not detect a false positive, that is to say, a violation in a term

when such term is not valid for the evaluation of the SLA. Test requirements TR4 are included in this category. They represent situations where both the Qualifying Condition and the Service Level Objective are not satisfied so the monitoring system must be aware that this term is inapplicable and it cannot be evaluated as violated.

Category 4 (C4) includes those requirements where a service associated to a Guarantee Term is not executed so the term must be evaluated as Not Determined. The fulfillment of these requirements may represent a problem during the evaluation process because there is a lack of information to determine whether a term is being fulfilled or not.. Due to this concern, these requirements are used to test whether the monitoring system is able to perform the evaluation process properly even when a service (method) has not been executed. Furthermore, these tests may lead to detect problems not in the application but in the SLA specification itself so the agreement can be reviewed and updated accordingly. Test requirements TR5 and TR6 are included in this category.

C. *Derivation of Test Cases*

The selection of the coverage level allows the tester to decide how thorough the SUT will be tested and, by means of the identification and categorization of test requirements, which situations are more interesting to test. Focusing on the Guarantee Term Coverage Level, we identify and categorize a set of test requirements for each of the Guarantee Terms specified in the SLA so the union of all of these requirements represents the final set of situations that must be exercised. The fulfillment of such test requirements is performed through the definition and executions of test cases. The derivation of these test cases aims at covering as many test requirements as possible with the most affordable cost. This involves that many test requirements can be covered using the same test case.

When deriving a test case, we are deciding which of the test requirements is going to be exercised during the execution of such test case. The same test case may exercise other test requirements obtained from different Guarantee Terms. Thus, bearing this fact in mind, the tester must decide how test requirements are combined with the aim of obtaining the smallest set of test cases that achieve coverage as higher as possible in this level.

V. CASE STUDY

In this section, we will illustrate the identification and categorization of test requirements over an eHealth service based application, which is used as a case study. This system was proposed within the context of the PLASTIC European project [19] and has been used in previous testing approaches [2][6]. The original example specifies the set of conditions in an SLA that should be satisfied by the constituent services of the eHealth system. We have added new conditions regarding functional features of the system in order to illustrate the identification of the test requirements. Furthermore, we have deleted the conditions that affect the availability of the services because we are considering conditions that affect only one execution of the SUT. The SLA we have used in this work can be downloaded from [7].

The behavior of the SUT is represented in Figure 2. Basically, the software under test is deployed as a composite service (WSHealth) that receives an alarm from a hospital and it must look for an available professional (WSDoctor or WSSupervisor) who will take responsibility for handling the incident. To do this, a service that manages the list of professionals of the hospital is queried (WSRegistry). This service provides a list of IP addresses for the professionals who are available at that moment depending on the type of the received alarm (*Emergency* or *Not Confirmation*). These professionals may be connected to the system through wired or mobile devices so the conditions related to both connections are different.

The SLA that governs the execution of this system contains 14 Guarantee Terms (GTs) related to 6 different services and 9 service methods. Twelve of the GTs have the whole general structure of a Guarantee Term, i.e., a Scope, Qualifying Condition and Service Level Objective. The other two GTs do not have a Qualifying Condition.
By applying the aforementioned identification procedure to this case study, we have identified 66 test requirements. All of these requirements have been classified according to the five categories described in Section IV. The results of the categorization are shown in Table II. In this table, the services and methods that constitute the case study are represented in the first two columns and their associated guarantee terms are indicated in the third column. The following columns show the number of test requirements identified in each category. Finally, the last column shows the total number of test requirements identified from each guarantee term.

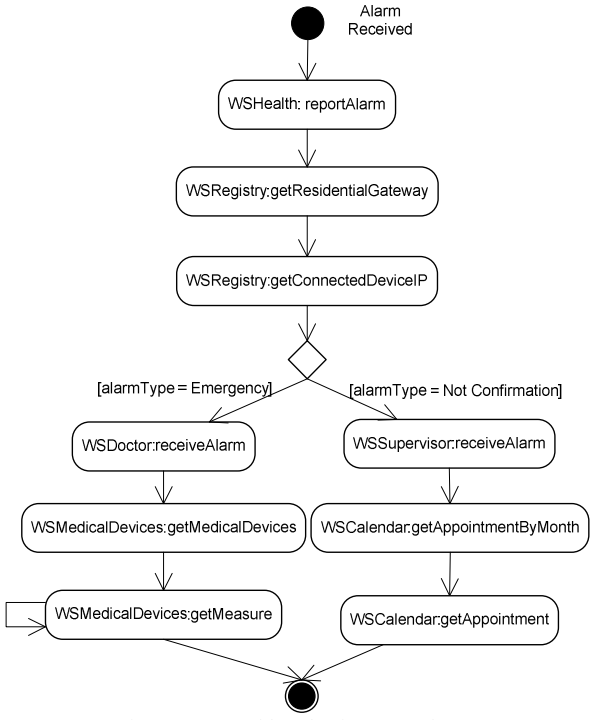

Figure 2. eHealth Behavior Example

TABLE II. *SLA TRACEABILITY SUMMARY*

Service	Method	GT	Categories					Test Req.
			C1	C2	C3	C3.1	C4	
WSHealth	reportAlarm	GT1	1	1	2	1	1	5
		GT2	1	1	2	1	1	5
WSRegistry	getResidentialGateway	-						-
	getConnectedDeviceIP	GT3	1	1	2	1	1	5
		GT4	1	1	2	1	1	5
		GT5	1	1	-	-	1	3
		GT6	1	1	2	1	1	5
		GT7	1	1	2	1	1	5
WSDoctor	receiveAlarm	GT8	1	1	2	1	1	5
		GT9	1	1	2	1	1	5
WSSupervisor	receiveAlarm	GT10	1	1	2	1	1	5
		GT11	1	1	2	1	1	5
WSMedicalDevice	getMedicalDevices	GT12	1	1	-	-	1	3
	getMeasure	GT13	1	1	2	1	1	5
		GT14	1	1	2	1	1	5
WSCalendar	getAppointmentByMonth	-	-	-	-	-	-	-
	getAppointment	-	-	-	-	-	-	-
TOTAL			14	14	24	12	14	66

In this case study, 14 requirements are included in category C1 (one for each guarantee term which are evaluated as Fulfilled), 14 requirements are included in category C2 (one for each guarantee term which are evaluated as Violated), 24 requirements are included in category C3 (two for each guarantee term that contains qualifying condition, which are evaluated as Inapplicable), 12 requirements are included in category C3.1 (one for each guarantee term where both the qualifying condition and the service level objective are not satisfied and, thus, the term is evaluated as Inapplicable) and, finally, 14 requirements are included in category C4 (one for each guarantee term where the methods of the services are not executed so as the term is evaluated as Not Determined). It is worthy mention that test requirements classified within C3.1 are also included in category C3 so the number of test requirements showed in the last column is exactly the addition of the numbers represented in the columns C1, C2 and C3 and C4.

Below we detail the identification of test requirements from two of the Guarantee Terms specified in the SLA (Figure 3). This pair of terms is related to the same service, which is in charge of providing the list of available professionals at the time of the alarm arrival. The first GT specifies the temporal threshold for the service to give the response when the alarm type is an *Emergency* whereas the second one indicates that it is mandatory to find at least one available professional to manage the incident. The specification of the test requirements for these two Guarantee Terms is presented in Table III. The first column shows the guarantee term from whom the test requirements have been identified. The second and third columns list the services and methods each guarantee term applies to. The fourth column numbers unequivocally the test requirement. The fifth and sixth columns specify the identifiers of the generic test requirement and the category. The seventh column specifies the content of the test requirement. The last two columns represent the expected behavior of the SUT regarding the SLA, including evaluation value of each guarantee term and the consequences derived from such evaluation.

Regarding the Guarantee Term GT3 of the SLA and according to the procedure described in Section IV, four test requirements represent the potential situations where the fulfillment/violation of the Qualifying Condition and the Service Level Objective of *GT3* are combined (TR1-TR4). These test requirements are identified with numbers 11-14. In this GT3, particular case PC2 is applied so, instead of identifying TR5 and TR6, just one test requirement is identified (number 15) and it represents that the method of the service specified in the Scope is not executed.

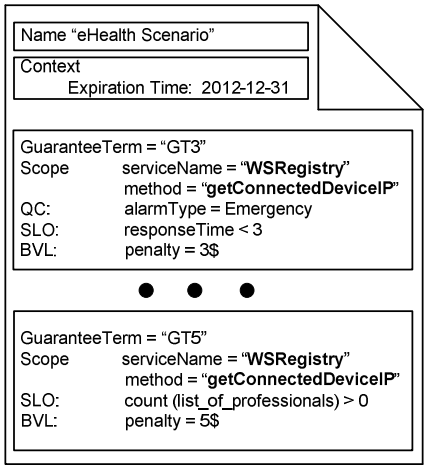

Figure 3. Excerpt of the SLA

TABLE III. *EXCERPT OF THE TEST REQUIREMENT SPECIFICATION*

GT	Service	Method	N	ID	Cat	Test Requirement	Expected Behavior	
							Ev. Value	Consequences
...								
GT3	WSRegistry	getConnectedDeviceIP	11	TR1	C1	alarmType == Emergency & responseTime < 3	Fulfilled	
			12	TR2	C2	alarmType == Emergency & responseTime ≥ 3	Violated	Penalty = 3$
			13	TR3	C3	alarmType != Emergency & responseTime < 3	Inapplicable	
			14	TR4	C3 – C3.1	alarmType != Emergency & responseTime ≥ 3	Inapplicable	
			15	TR5	C4	The method getConnectedDeviceIP of WSRegistry is not executed	Not Determined	
...								
GT5	WSRegistry	getConnectedDeviceIP	21	TR1	C1	count (list_of_professionals) > 0	Fulfilled	
			22	TR2	C2	count (list_of_professionals) ≤ 0	Violated	Penalty = 5$
			23	TR5	C4	The method getConnectedDeviceIP of WSRegistry is not executed	Not Determined	
...								

Regarding GT5, particular case PC1 is applied. On the one hand, this involves the identification of test requirements numbers 21-22 because the term does not include Qualifying Condition. On the other hand, it allows joining TR5 and TR6 into the test requirement number 23. The fulfillment of the first of these test requirements (number 21) aims at checking whether the application is able to find an available professional at the time of receiving an alarm. The fulfillment of test requirement number 22 tests the behavior of both the application and the monitoring system when there is not any professional available to manage the alarm. On the other hand, test requirement number 23 tests the behavior of the system when the invocation of the registry is not performed.

As can be seen, this procedure may identify test requirements that represent equivalent situations (see test requirements numbers 15 and 23). However, this fact will be considered when deriving the test suite because both test requirements will be exercised through the design and execution of a unique test case.

Finally, in the last column of Table III we have only represented the consequences derived from the test requirements numbers 12 and 22 according to the specification of the Guarantee Terms. When deriving the test cases, the tester will describe the full expected output of such test case bearing in mind the information about the behavior of the SUT.

VI. RELATED WORK

In the scope of Service Oriented Architectures, much effort is being focused on the detection of SLA violations using different approaches. Basically, these works may be categorized in two main groups: in the first group we include such works which aim at detecting these violations at runtime when the SUT is already deployed in its operational environment; the second category includes the group of approaches that try to anticipate the detection of problems or event the prevention of them, before these problems lead to consequences for the stakeholders.

Regarding the first group, several works have addressed the testing of SLAs using monitoring approaches to detect SLA violations. Mahbub and Spanoudakis [13] propose to model and monitor the conditions specified in WS-Agreement using an Event Calculus (EC) based approach. Raimondi et al. [20] proposed a system that automatically monitors SLAs, translating timeliness constraints into timed automata, which is used to verify traces of services executions. Comuzzi et al. [4] tackles the relation between the establishment and monitoring of SLAs in the scope of SLA@SOI European Project. In addition to these works, other different systems have been developed to monitor service based applications with the aim of detecting SLA violations, for example, SALMon [15], SLAMonitor [8] or CLAM [3].

Monitoring techniques have also been used to recollect information of the SUT in order to prevent SLA violations. Leitner et al. [10] propose a framework that allows monitoring and predicting SLA violations before they have occurred using machine learning techniques. Ivanovic et al. [9] propose a constraint based approach to monitor and analyze the QoS metrics included in the SLA for the purpose of anticipating the detection of potential SLA violations. Schmieders et al. [21] combined monitoring and prediction techniques in order to prevent SLA violations. Lorenzoli and Spanoudakis [11] presented EVEREST+ framework which supports the monitoring and prediction of potential violations of the QoS metrics specified in the SLA.

Finally, few approaches have addressed the identification of tests from the SLA specification with the aim of anticipating the detection of violations. Di Penta et al. [5] perform black-box and white-box testing using Genetic Algorithms to detect SLA violations in service compositions and Palacios et al. [16] propose to test the conditions of the SLA specified in WSAG using a well known testing technique, the Category Partition Method. Furthermore, Bertolino et al. [2] propose the PUPPET framework, which allows generating stubs from the WSAG, WSDL and BPEL specification of the services to test SLA-aware service compositions.

VII. Conclusions and Future Work

In this paper we tackle the testing of SLA-aware service-based applications with three main contributions. We have devised different coverage levels regarding the SLA that allow deciding or establishing the thoroughness of the tests. Focusing on the first of these levels (*Guarantee Term Coverage Level*), we have proposed a way to evaluate Guarantee Terms of SLAs taken the syntactic and semantic structure of such terms into account. Using this logic as a foundation, we have proposed a systematic procedure to identify a set of test requirements for SLA Guarantee Terms achieving a full coverage within the GT Coverage Level. These test requirements may be exercised through the generation of a suitable set of test cases or used to support the derivation of a monitoring plan to observe and check the SLA Guarantee Terms at runtime. The identification method has been applied to a case study of an e-Health system that was originally proposed by the European project PLASTIC.

In our future work, we will focus on improving the coverage criteria, using the information represented in the logical and hierarchical structure of the SLA (*Compositor* and *SLA Coverage Levels*). From the final set of test requirements, we will have to evaluate the testability of such test requirements in order to determine which of the requirements can be exercised through the execution of tests and which of them should be checked at runtime. For those executable test requirements, we will be able to provide the tester with guidelines that allow deriving a suitable test suite which aims at exercising as many test requirements as possible maximizing the trade-off cost-benefit.

Finally, the task of obtaining the test situation is currently performed manually by the tester. As WS-Agreement is an XML-based language, the identification of these situations can be performed automatically so we will study the feasibility of developing a tool that automates this process in the future.

Acknowledgment

This work has been partially funded by the Department of Science and Innovation (Spain) and ERDF funds within the National Program for Research, Development and Innovation, project Test4DBS (TIN2010-20057-C03-01) and FICYT (Government of the Principality of Asturias) Grant BP09-075.

References

[1] A. Andrieux, K. Czajkowski, A. Dan, K. Keahey, H. Ludwig, T. Nakata, J. Pruyne, J. Rofrano, S. Tuecke, M. Xu, “Web Services Agreement Specification”, 2007.

[2] A. Bertolino, G. De Angelis, L. Frantzen, A. Polini, “Model-based generation of testbeds for web services,” Proc of Testcom/FATES, Lecture Notes In Computer Science, vol. 5047. Springer-Verlag, Berlin, Heidelberg, 2008, pp. 266-282.

[3] K. Bratanis, D. Dranidis, A. J. H. Simons, “SLAs for cross-layer adaptation and monitoring of service-based applications: a case study,” In Proceedings of the International Workshop on Quality Assurance for Service-Based Applications (QASBA), 2011.

[4] M. Comuzzi, C. Kotsokalis, G. Spanoudakis, R. Yahyapour, “Establishing and Monitoring SLAs in Complex Service Based Systems,” Proc. IEEE International Conference on Web Services (ICWS), 2009, Los Angeles, CA.

[5] M. Di Penta, G. Canfora, G. Esposito, V. Mazza, M. Bruno, “Search-based testing of service level agreements,” Proc. 9th Annual Conference on Genetic and Evolutionary Computation (GECCO 07), London, ACM, New York, 2007, pp. 1090-1097.

[6] L. Frantzen, M. N. Huerta, Z. G. Kiss, T. Wallet, “On-The-Fly Model-Based Testing of Web Services with Jambition,” in 5th Int. Workshop on Web Services and Formal Methods (WS-FM 2008), ser. LNCS, no. 5387. Springer, 2009, pp. 143-157.

[7] Software Engineering Research Group (GIIS) homepage: http://giis.uniovi.es/testing/downloads/?lang=en (accessed February 2012)

[8] N. Goel, N.V.N. Kumar, R.K. Shyamasundar, "SLA Monitor: A System for Dynamic Monitoring of Adaptive Web Services," Proc. 9th IEEE European Conference on Web Services (ECOWS), 2011, pp.109-116.

[9] D. Ivanovic, M. Carro, M. Hermenegildo, “Constraint-Based Runtime Prediction of SLA Violations in Service Orchestrations,” Proc. International Conference on Service Oriented Computing (ICSOC), 2011, pp. 62-76.

[10] P. Leitner, A. Michlmayr, F. Rosenberg, S. Dustdar, “Monitoring, prediction and prevention of SLA violations in composite services,” Proc. IEEE International Conference on Web Services (ICWS) Industry and Applications Track, 2010, pp.369-376.

[11] D. Lorenzoli, G. Spanoudakis, “EVEREST+: Runtime SLA Violations Prediction,” 5th Middleware for Service-oriented Computing Workshop, in conjunction with the 11th ACM/IFIP/USENIX International Middleware Conference, 2010.

[12] D. Lorenzoli, G. Spanoudakis, “Runtime Prediction of Software Service Availability,” Int. Conference on Software Engineering Research and Practice (SERP'11), July 18-21, 2011, USA.

[13] K. Mahbub, G. Spanoudakis, “Monitoring WS-Agreements: an event calculus based approach,” in Test and Analysis of Service Oriented Systems, Springer V., 2007, pp. 265-306.

[14] J. Offut, L. Nan, P. Ammann, X. Wuzhi, “Using abstraction and Web applications to teach criteria-based test design," 24th IEEE-CS Conference on Software Engineering Education and Training (CSEE&T), 2011, pp.227-236.

[15] M. Oriol, J. Marco, X. Franch, D. Ameller, “Monitoring Adaptable SOA System using SALMon,” Workshop of Service Monitoring, Adaptation and Beyond (MONA+), ServiceWave Conf., 2008.

[16] M. Palacios, J. García-Fanjul, J. Tuya, C. de la Riva, “A proactive approach to test service level agreements,” Proc. Fifth International Conference on Software Engineering Advances (ICSEA), 2010, pp. 453-458.

[17] M. Palacios, J. García-Fanjul, J. Tuya, “Testing in service oriented architectures with dynamic binding: A mapping study,” Information and Software Technology, vol. 53 (3), March 2011, pp. 171-189.

[18] M. Palacios, “Defining an SLA-aware method to test service-oriented systems,” Proc. 9th International Conference on Service Oriented Computing (ICSOC), PhD Symposium, Dec. 2011.

[19] PLASTIC European Project homepage: http://www.ist-plastic.org/ (accessed February 2012).

[20] F. Raimondi, J. Skene, W. Emmerich, “Efficient Online Monitoring of Web-Service SLAs,” Proceedings of the 16th ACM SIGSOFT Int. Symposium on Foundations of Software Engineering (SIGSOFT'08/FSE-16), 2008.

[21] E. Schmieders, A. Micsik, M. Oriol, K. Mahbub, R. Kazhamiakin, “Combining SLA prediction and cross layer adaptation for preventing SLA violations,” Proc. 2nd Workshop on Software Services: Cloud Computing and Applications based on Software Services, 2011, Timisoara, Romania.